САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Утицын Олег Анатольевич

Выпускная квалификационная работа бакалавра

Реализация поддержки мобильных устройств для Web ГИС

Направление 010300

«Фундаментальная информатика и

информационные технологии»

Научный руководитель,

ст. преп.

Соловьев П.А.

Рецензент,

кандидат с.-х. наук

Афонин А.Н.

Санкт-Петербург

2017

Содержание

Терминология

Сервер — программный компонент вычислительной системы, который выполняет набор функций, связанных с обработкой и представлением данных, запрошенных клиентом.

Клиент — программный компонент вычислительной системы, который посылает запросы серверу и отображает различного рода данные в браузере.

Браузер — программное обеспечение, предназначенное для отображения веб-страниц и визуализации данных, приходящих с сервера.

API (от англ. *Application Programming Interface*) — интерфейс программирования приложений, предоставляющий набор готовых классов и методов, который поставляется в виде библиотек или сервисов.

REST (от англ. *Representational State Transfer*) — стиль архитектуры приложения, описывающий взаимодействие клиента и сервера.

Узел — компьютерное устройство, которое является частью компьютерной сети.

Протокол — набор соглашений, определяющих обмен данными между различными программами.

DOM (от англ. *Document Object Model*) — представление веб-страницы в виде дерева объектов, которое позволяет получать или изменять содержимое страницы посредством языка программирования.

Фреймворк — набор всевозможных программных инструментов, представляющий некий каркас архитектуры приложения и облегчающий разработку программ.

OGC (от англ. *Open Geospatial Consortium*) — международная, добровольная организация по разработке стандартов в области геоинформационных сервисов.

Проекция — способ отображения криволинейной поверхности на плоскость.

Перепроецирование — математическое преобразование картографических данных из одной проекции в другую.

Растровая реклассификация — операция, которая преобразует растровое изображение посредством матрицы наложения, в соответствии с которой замещаются значения на данном растре.

Растровая алгебра — операция, принимающая в качестве своих аргументов формулу и набор растровых слоев, значения точек которых будут преобразованы в соответствии с формулой.

Плагин — программный модуль, подключаемый к программе, который предназначен для расширения ее возможностей.

Компиляция — преобразование программы, написанной на языке высокого уровня, в программу, составленной на низкоуровневом языке.

Компилятор — программа, выполняющая компиляцию.

СУБД (*Система Управления Базами Данных*) — совокупность программных средств, позволяющих создавать, поддерживать, контролировать, изменять базы данных.

HTML (от англ. *HyperText Markup Language*) — язык разметки, предназначенный для представления документов в сети Интернет.

CSS (от англ. *Cascading Style Sheets*) — язык стилей, предназначенный для описания внешнего вида html-документа.

URL (от англ. *Uniform Resource Locator*) — уникальный идентификатор ресурса в сети Интернет.

Введение

Автору данной работы было предложено принять участие в доработке открытой многопользовательской географической информационной системы (далее ГИС) O-GIS, находящейся по адресу http://app.o-gis.org/.

Данная система предназначена для решения задач анализа и обработки пространственных данных более чем одним пользователем в реальном времени. Система нацелена на работу с растровыми или векторными слоями, из которых можно составлять композиции слоев, получать информацию о конкретной точке текущей композиции, а также применять операции реклассификации и растровой алгебры. Преимуществами такой системы являются единое информационное пространство и вычисление ресурсоемких операций на стороне сервера, благодаря которым пользователи могут осуществлять работу над одним проектом из разных точек нашей планеты и использовать устройства с малыми вычислительными мощностями, которые имеют доступ в Internet.

Конечной целью при разработке проекта O-GIS было создание приложения, которое функционирует не только на стационарных, но и на мобильных устройствах. В первую очередь это обусловлено тем, что в большинстве случаев система применяется в условиях, не предназначенных для настольных компьютеров, — «в поле», где доступны только те устройства, которые имеют достаточно малые размеры и которые не зависят от проводной сети.

На текущий момент работа на мобильных устройствах ограничена и неудобна — элементы системы не адаптированы под мобильные устройства; редактор картографических данных не помещаются во всю ширину экрана, в связи с чем интерфейс приходится масштабировать и перемещаться по нему как в горизонтальном, так и в вертикальном направлении; функциональность текущей версии O-GIS урезана — например, невозможно изменять порядок отображения слоев, перетащив имя слоя вверх либо вниз в окне настроек

редактора. Также на стороне клиента используется устаревшая библиотека визуализации картографических данных OpenLayers 2 [1]. На момент написания выпускной работы актуальна 4-ая версия — она направлена на повышение производительности, улучшение визуальной составляющей и API.

Цель выпускной квалификационной работы состоит в следующем: необходимо модернизировать систему O-GIS таким образом, чтобы использование редактора на мобильных устройствах не вызывало трудностей, перечисленных выше. Отметим, что замена устаревшей библиотеки визуализации картографических данных OpenLayers 2 будет логичным этапом модернизации системы — прирост производительности существенно важен, в особенности на устройствах, мощность которых пока еще мала по сравнению с современным персональным компьютером.

В этой связи первая часть данной работы «Обзор библиотек визуализации картографических данных» направлена на сравнение современных библиотек визуализации картографических данных на стороне клиента и выбор наиболее подходящей из них для O-GIS.

Решение задачи адаптированности системы под мобильные устройства обсуждается во второй главе выпускной квалификационной работы.

Постановка задачи

Задача выпускной квалификационной работы заключается в следующем: необходимо модифицировать редактор композиций проекта O-GIS до состояния, в котором использование системы на таких мобильных устройствах, как планшеты и смартфоны, будет осуществляться без ограничений на функциональную составляющую, будет удобным пользователю, и, возможно, будет превосходить в скорости предыдущую версию системы.

К модифицированному модулю O-GIS выдвигаются следующие требования:

1. Элементы интерфейса системы, предназначенные для настройки редактора композиции, визуализации слоев должны:
   A. Располагаться в удобной зоне экрана — так, что бы пользователь мог быстро взаимодействовать с системой.
   B. Иметь оптимальные размеры в зависимости от размера экрана — не слишком мелкие, что затруднит выбор нужного элемента управления, и не слишком крупные, что займет большую часть экрана.
   C. Быть адаптивными — размеры и позиция элементов должны меняться при изменении размеров экрана браузера, а сами элементы — занять при этом оптимальное положение на экране.
   D. Быть интерактивными — элементы должны реагировать на действия пользователя, причем их реакция должна иметь одинаковый результат на различных устройствах. Например, событие клика мышкой по элементу должно быть эквивалентным событию касания элемента пальцем на сенсорном экране.
   E. Иметь соответствующий внешний вид, отражающий специфику операции, которая выполняется при взаимодействии с данным элементом.

F. Иметь асинхронные обработчики событий — страница проекта не должна перезагружаться при взаимодействии пользователя с элементами управления.
G. Быть атомарными. Элемент назовем атомарным, если его нельзя разбить на составляющие части, которые будут выполнять независимые операции.

2. Окно, содержащее элементы управления редактором, должно:
   A. Иметь функцию сворачивания.
   B. Сворачиваться по умолчанию, если экран устройства, на котором запущен проект, достаточно мал для размещения всех элементов интерфейса.
   C. Осуществлять функцию изменения порядка слоев, как на мобильном, так и на стационарном устройстве.
3. Окно отображения композиции слоев должно осуществлять функции масштабирования, навигации, геокодирования как на сенсорном устройстве, так и на персональном компьютере.
4. Система должна предоставлять пользователю возможность выполнения стилизации, перепроецирования, растровой реклассификации, растровой алгебры слоев. Данные операции реализованы в актуальной O-GIS, однако их выполнение на мобильных устройствах трудновыполнимо — пользователю приходится масштабировать экран и перемещаться по нему в горизонтальном и вертикальном направлениях, поскольку окна, отвечающие за такие возможности не помещаются на экран.

Если при решении описанной выше задачи возникнут непреодолимые трудности, автору данной работы следует привести обоснования, почему улучшение текущей реализации O-GIS в указанных направлениях осуществить невозможно.

Обзор библиотек визуализации картографических данных

Библиотека, позволяющая визуализировать картографические данные, является важным компонентом архитектуры приложения O-GIS, поскольку от нее зависят такие параметры, как скорость загрузки данных, их отрисовка и скорость отклика карт на действия пользователя. В этой связи такой компонент является узким местом в системе — выбор библиотеки будет влиять прямым образом на производительность, что является одним из главных требований к адаптированной под мобильные устройства O-GIS. Как уже было сказано выше, текущая библиотека отображения пространственных данных, используемая в актуальной O-GIS, устарела — последний выпуск состоялся 9 июля 2013 года [2]. Однако ее до сих пор поддерживают, о чем говорится на официальном сайте продукта [1], но реализация новых возможностей и улучшение визуализации карт, производительности и повышение уровня абстракции кода осуществляется четвертой — последней версией OpenLayers [3]. Отметим, что OpenLayers не является единственным продуктом в данной области, существуют альтернативные решения — например библиотека LeafLet [4], написанная украинским разработчиком Владимиром Агафонкиным [5].

В связи с вышеизложенным первая часть задачи данной работы будет состоять в выборе наиболее оптимальной JavaScript библиотеки визуализации картографических данных для O-GIS.

Критерии сравнения библиотек

Были выдвинуты следующие критерии для сравнения библиотек:

1. Возможность управления коллекцией отображаемых слоев:
   A. Сложность операции инициализации карты, т.е. первоначальной загрузки карты для дальнейшего отображения слоев.
   B. Сложность операции добавления слоя с настройками визуализации по умолчанию.

C. Сложность операции добавления слоя с пользовательскими настройками визуализации, как то: размер карты, количество уровней масштабирования, положение центра карты, минимальный и максимальный масштаб карты, стилизация слоев.

D. Сложность операции изменения настроек визуализации слоя, перечисленных в пункте C.

E. Сложность доступа к конкретному слою из отображаемого набора.

F. Сложность операции удаления слоя из карты.

2. Масштабирование карты.
3. Навигация по карте.
4. Размер библиотеки.
5. Вывод дополнительной информации на область карты: маркеры, всплывающие окна.
6. Является ли библиотека Open Source, т. е. доступен ли код проекта любому, желающему использовать ее в своих целях, разработчику.
7. Адаптированность под мобильные устройства.
8. Кросс-браузерность, т. е. поддержка библиотеки всеми популярными браузерам.
9. Экспорт карт на локальную машину.
10. Поддержка следующих форматов пространственных данных: WMS, WFS, KML, GeoJSON.
11. Наличие материалов для изучения библиотеки.
12. Поддержка геокодирования — отображения географических координат конкретной точки карты.
13. Поддержка основных проекций, т. е. способов отображения криволинейной поверхности (например, земного шара) на плоскость.

Сложность операции будем измерять как количество вызванных методов, необходимых для выполнения конкретного критерия.

Данные критерии были выбраны исходя из основных требований к проекту O-GIS:

- Использование свободно-распространяемого ПО.
- Возможность единовременной работы с композицией слоев (их добавление, удаление, изменение порядка отображения).
- O-GIS как web-приложение:
    - Централизованная генерация оперативных карт, имеющих срок жизни не более 24 часов.
    - Использование O-GIS на мобильных устройствах в связи с взаимодействием с системой «в поле».
    - Обмен информацией, в том числе объектами O-GIS, между различными пользователями системы (ГИС как социальная сеть).
    - Поддержка O-GIS популярными web-браузерами.
- Понятный пользовательский интерфейс.

В данной главе будет представлено сравнение четырех наиболее распространенных и хорошо поддерживаемых библиотек визуализации картографических данных: Google Maps, Polymaps, Leaflet и OpenLayers 4.

Google Maps

Google Maps является самой старой (создана в феврале 2005 года [6]) из рассматриваемых библиотек и самым простым в использовании веб-инструментом для отображения картографических данных. Она используются в миллионах приложений и на многих сайтах, которым необходимо передать информацию пользователю с помощью географических карт.

Для быстрой загрузки и хорошей совместимости с мобильными устройствами программисты Google Maps разработли JavaScript API. При создании этого API особое внимание было уделено разработке приложений для современных мобильных устройств на платформах Android и IOS. API предоставляет широкую функциональность: масштабирование карт; перемещение по карте с помощью мыши и клавиатуры на персональном компьютере, а на сенсорном устройстве — с помощью касаний экрана; выбор типа карты: дорожные — установлены по умолчанию, спутниковые — отображение снимков со спутников Земли, физические — основаны на данных о ландшафте; добавление на кару маркеров, html-элементов, линий, полигонов, изображений; гибкая настройка обработчиков событий, возникающих при взаимодействии с картой; поддержка маршрутизации — генерирование оптимальных маршрутов из одного пункта карты в другой [7].

Рассмотрим основные операции Google Maps API.

Инициализация карты выполняется с помощью конструктора `Map`:

```
new google.maps.Map(mapDiv, MapOptions)
```

Обозначения:

- `mapDiv` — обязательный параметр — ссылка на html-элемент, внутри которого будет создана карта;
- `MapOptions` — объект, два параметра которого являются обязательными — `center` и `zoom`, устанавливающие центр, например, в формате `LatLng` (класс, с помощью которого задаются географические координаты: широта и долгота) и первоначальный уровень масштабирования соответственно. Дополнительные параметры — цвет фона карты, максимальный, минимальный уровни масштабирования, либо полное его отключение, включение функции вращения карты и т. п.

Алгоритм инициализации карты следующий:

1. Добавление тега `script` на html-страницу, которая будет отображать карту. Тег должен содержать поле `key` — персональный ключ, который можно получить при регистрации своего приложения в системе Google.
2. Создание html-элемента для хранения карты (например — `div`).
3. Указание высоты созданного html-элемента.
4. Определение функции JavaScript, которая и инициализирует карту в html-элементе.

В библиотеке Google Maps существует возможность добавления собственных слоев в формате GeoJSON, KML и GeoRSS.

Подробно разберем работу с форматом GeoJSON — для других форматов действия будут аналогичными. Для добавления на карту слоя такого формата необходима всего лишь одна операция:

```
map.data.loadGeoJson('google.json')
```

Обозначения:

- `map` — ссылка на экземпляр класса Map, созданный на шаге инициализации карты;
- `map.data` — объект для хранения произвольных геопространственных данных;
- `loadGeoJSON()` — метод для добавления слоя в формате GeoJSON;
- `google.json` — путь до файла в формате GeoJSON.

Если требуется изменить стиль по умолчанию для всех слоев, понадобится еще одна операция:

```
map.data.setStyle(StyleOptions)
```

Обозначения:

- `StyleOptions` — объект параметров, устанавливающих цвет заливки,

прозрачность, видимость, порядок отображения слоев, и включающих обработчики на конкретные события, посредством которых пользователь будет взаимодействовать с картой и т. п.

Данная операция добавляет указанные стили ко всем слоям, находящимся на карте. Чтобы метод `setStyle()` устанавливал стили для каждого наложения на карте, необходимо вместо объекта `StyleOptions` передать функцию, аргументом которой будет являться конкретный слой, и, в зависимости от обрабатываемого слоя, стили можно устанавливать индивидуально, возвращая объект настроек для каждого. Также можно изменить стиль для конкретного слоя посредством вызова метода `overrideStyle()`, первым аргументом которого является слой, к которому нужно добавить стили или изменить их, вторым аргументом — объект `StyleOptions`, описанный выше.

Для удаления всех стилей следует вызвать метод `setStyle()` без параметров. Для удаления конкретного слоя используется метод `remove()`, аргументом которого является слой, который нужно удалить с карты.

В Google Maps API для создания карт из географических данных используется проекция Меркатора (EPSG:3857— код в официальном онлайн-реестре, хранящем данные о проекциях [8]). Меридианы и параллели в такой проекции представляются параллельными линиями. Расстояние между меридианами одинаковое, а между параллелями — увеличивается при перемещении от экватора к полюсам. Отсюда следует, что масштаб на карте в этой проекции не является постоянным — он также увеличивается от экватора [9].

К сожалению, функциональность Google Maps ограничена: библиотека привязана к конкретной карте Google, а так же использует единственную проекцию, что неприемлемо для системы O-GIS. К тому же Google Maps не является библиотекой с открытым исходным кодом — невозможно заглянуть внутрь библиотеки и узнать, как она функционирует, а также — изменять код под свои требования.

Leaflet

Leaflet является JavaScript библиотекой визуализации пространственных данных с открытым исходным кодом. Она эффективно работает на всех основных настольных и мобильных платформах, может быть расширена большим количеством плагинов, имеет простой в использовании и хорошо документированный API [4]. Leaflet поддерживает форматы слоев WMS, GeoJSON, но не поддерживает WFS, GML. Поддержка формата KML осуществляется с помощью отдельного модуля, разработанного для библиотеки [10].

Для инициализации карты с помощью библиотеки Leaflet, достаточно выполнить следующие шаги:

1. Подключить CSS стили и JavaScript скрипт на html-странице.
2. Создать html-элемент, в котором будет отображаться карта (например — `div`).
3. Настроить высоту созданного html-элемента.
4. Определить JavaScript функцию, которая и будет инициализировать карту:

```
L.map('mapId', options).setView(center, zoom)
```
(1)

Обозначения для операции 1:

- `L.map` — метод, генерирующий экземпляр класса `Map` (в данном случае `map` — это фабричный метод, т. е. создание экземпляра класса инкапсулировано в этом методе);
- `mapId` — идентификатор html–элемента, созданного на шаге 2;
- `options` — необязательный параметр — объект различных настроек карты;
- `setView` — метод, отображающий карту в определенном положении;
- параметры `center` и `zoom` — массивы, содержащие координаты широты и долготы, и начальный масштаб карты соответственно.

На самом деле параметр `center` можно задавать с помощью класса `L.LatLng`, передав ему географические координаты, но использование массива — это «синтаксический сахар» — так намного короче и понятнее, далее они будут преобразованы в экземпляр этого класса самостоятельно. Такое уточнение справедливо и для задания области карты — ее также можно задавать неявно.

Заметим, что в отличие от Google Maps API на данном этапе карта только инициализируется, она не содержит никаких слоев и данных — их добавление ложится на плечи разработчика. Очевидно, что такая гибкость является плюсом библиотеки.

Рассмотрим основные операции, используемые для манипулирования слоями по протоколу WMS. Для добавления слоя на карту необходимо вызвать следующую цепочку методов:

```
L.tileLayer.wms(url, options).addTo(map)
```

Обозначения:

- `wms()` — метод для загрузки слоев по протоколу WMS;
- `url` — адрес загрузки слоя;
- `addTo()` — метод, который отображает слой на карте;
- `map` — переменная, в которой хранится ссылка на карту, созданную на шаге инициализации;
- `options` — объект настроек, с помощью которого можно установить:
  - `layers` — список слоев, которые нужно отобразить (тип данного поля строковый, поэтому требуется разделять имена слоев запятыми);
  - `format` — формат изображения WMS (например — «image/jpeg»);
  - `styles` — список стилей, разделенных запятыми. Позиция стиля определяется исходя из позиции слоя в строке `layers`, к которому требуется применить данный стиль;

- `crs` — система координат, которая определяет как будут проецироваться географические координаты на экран;
- также можно указать минимальный, максимальный масштаб, его уровни, прозрачность и `zIndex` — порядок отображения слоя на карте.

Для установки новых параметров слоев, необходимо вызвать следующий метод:

```
setParams(params, noRedraw)
```

Обозначения:

- `params` — объект, идентичный объекту `options` в методе `wms()`;
- `noRedraw` — если равен false, отправляет повторный запрос к серверу и перерисовывает все слои.

Для манипуляции с несколькими слоями в библиотеке Leaflet используются группы. Следующая последовательность операций задаст группу из двух слоев, добавит к этой группе третий слой, и отобразит их на карте:

```
L.layerGroup([layer1, layer2]).addLayer(layer3).addTo(map)
```

Группы позволяют сокращать количество операций: их можно применять не к каждому слою, а единственный раз — к самой группе. Для доступа к конкретному слою из группы необходимо вызвать метод `getLayer(id)`, где `id` — это внутренний идентификатор слоя в группе.

Для получения всех слоев группы можно использовать метод `getLayers()`, который возвращает массив с элементами-слоями.

Удалить слой из группы можно вызовом метода `removeLayer(id)`. А для удаления слоя с карты — `layer.remove()`, где `layer` — ссылка на слой, который следует удалить.

Для пользователей по умолчанию будет доступна следующая функциональность: перемещение по карте с помощью мыши, изменение масштаба

двойным кликом, переход к области с помощью выделения нужного участка (при зажатой кнопке shift), навигация с помощью клавиатуры. Если устройство, на котором используется карта Leaflet — сенсорное, перемещаться по карте, изменять масштаб можно пальцами.

Также по умолчанию Leaflet использует сферическую проекцию Меркатора — EPSG:3857. В данной библиотеке поддерживаются следующие проекции: эллиптическая проекция Меркатора (EPSG:3395) — используется такими сервисами как Космоснимки [11], Яндекс карты [12]; проекция EPSG:4326 — точки полюсов которой обращены в линии; чем дальше осуществляем перемещение от экватора, тем сильнее любой объект на карте оказывается сплюснут по вертикали и растянут по горизонтали. Используют ее в том случае, когда преобразования географических координат в координаты, отображаемые на карте, должны быть тривиальны. Минус данной проекции очевиден — полярные территории становятся сплюснутыми. Также поддерживается Simple проекция, используемая при отображении картинок, карт игр, и т. д., то есть объектов, расположенных на плоскости. Проекцию можно задать с помощью поля объекта `options`, который был описан выше. Например, чтобы использовать проекцию EPSG:4326, в `options` нужно указать поле `crs`, со значением `L.CRS.EPSG4326`.

Библиотека Leaflet проста в использовании и предполагает отображение пространственных данных с базовой функциональностью, как то: навигация, масштабирование, вращение, стилизация и т. п., но для задач более высокого уровня приходится подбирать и устанавливать соответствующие модули, импортируемые с Leaflet, которые могут быть разработаны сторонними программистами, либо их может не существовать вообще. Для того, чтобы со временем было реально добавлять дополнительные возможности к проекту O-GIS, его разработчики должны быть уверены в том, что реализация этих возможностей не приведет к замене библиотеки, выбранной в данной работе.

Polymaps

Polymaps это свободно-распространяемая библиотека JavaScript для создания динамичных интерактивных карт в современных веб-приложениях. Данная библиотека ориентирована на работу с векторными данными значительных объемов. Это достигается за счёт использования языка разметки масштабируемой векторной графики — SVG и автоматического изменения детализации геометрии при переходе на другой масштабный уровень. Polymaps поддерживает два формата векторных данных — SVG и GeoJSON [13].

Для инициализации карты Polymaps и последующего отображения слоев необходимо выполнить следующие действия:

1. Подключить библиотеку Polymaps на html-странице.
2. Записать глобальный объект `polymaps` в локальную переменную `po`:

```
var po = org.polymaps
```

Это необязательный шаг, нужен для удобства дальнейших вызовов.

3. Выполнить три операции:

A. `var map = po.map();`

B. `map.container(document.body.appendChild(po.svg("svg")));`

C. `map.add(po.layer(load));`

Операция A получает экземпляр класса `Map` (с помощью фабричного метода `map`). Операция B строит в DOM-дереве элемент `svg` с именем `"svg"`, в котором будет инициализирована карта. Операция C добавляет слой на карту, используя переданную ей функцию `load` в качестве аргумента. Функция `load` получает объект координат, ключом которого является уникальная строка, основанная на координатах вида `“{Z}/{X}/{Y}”`, где `Z` — это уровень масштаба, а `X` и `Y` — координаты так называемого тайла.

Тайлы — это небольшие изображения одинаковых размеров, которые

служат фрагментами одной картинки. Преимущество тайлов заключается в том, что изображения малых размеров быстрее передаются по сети, а если их передавать параллельно — прирост производительности улучшится в разы: пользователю будет намного комфортней работать, когда карты загружаются моментально.

Значением вышеописанных ключей являются объекты, которые содержат информацию о координатах и уровне масштаба тайла. Функция `load` отвечает за размещение тайла на карте и реализуется программистом.

Все методы в Polymaps возвращают экземпляр класса `Map` — такое соглашение позволяет использовать цепочки методов. По умолчанию созданная карта не интерактивна, что бы исправить это, нужно вызвать метод `map.add(po.interact())`, который добавит следующие функции: перемещение по карте, изменение масштаба карты и генерирование событий при взаимодействии с картой.

В Polymaps существуют методы, с помощью которых можно изменить параметры карты:

- `map.size(x)` — меняет размер карты, где `x` — объект, в котором задается ширина и длина карты (поля `x` и `y` соответственно);
- `map.zoom(x)` — устанавливает уровень масштабирования, где `x` — положительное число;
- `map.center(x)` — меняет центр карты, где `x` — объект с географическими координатами нового центра.

Вышеуказанные методы также являются и геттерами — если их вызывать без аргументов, для каждого метода будет возращен соответствующий параметр карты.

Если возникнет потребность получить конкретный слой с карты, необходимо вызвать метод `layer.id(id)`, передав ему соответствующий идентификатор слоя.

Для изменения стиля слоя потребуется следующий метод:

```
attr(key, value)
```

Параметр `key` — это конкретное свойство, которое необходимо поменять, `value` — новое значение этого свойства. Например, вызов метода `attr("opacity", 0.5)` установит значение яркости слоя равным 0,5.

Для удаления слоя с карты используется метод `map.remove(x)`, где `x` — слой, который необходимо удалить.

Библиотека Polymaps проста в использовании и имеет высокую скорость отрисовки данных, однако она не подходит для визуализации для O-GIS, поскольку имеет ограничение на использование только векторных данных, а существенная работа в системе O-GIS будет производиться с растровыми изображениями.

OpenLayers 4

OpenLayers 4 является наиболее развитой библиотекой визуализации картографических данных с открытым исходным кодом, и, вероятно, широко используемой. Существует множество материалов в информационной глобальной среде, позволяющих быстро ознакомиться с методами OpenLayers 4 и использовать ее в своих приложениях.

OpenLayers 4 имеет объем 465 кб [14] по сравнению с прямым конкурентом — LeafLet, объем которого 154 кб [15], следовательно первоначальная загрузка web-приложения O-GIS окажется быстрее в случае выбора библиотеки LeafLet, но используя высокоскоростное интернет соединение, которое доступно каждому пользователю на сегодняшний день, данное преимущество будет незначительным.

В данной библиотеке разрешено использовать проекции EPSG:4326 и EPSG:3857, которые были описаны выше. Для реализации карт в других проекциях необходимо вначале создать экземпляр требуемой проекции, исполь-

зуя класс `ol.proj.Projection()`, а затем зарегистрировать ее с помощью метода `ol.proj.addProjection()`.

Для инициализации карты OpenLayers 4 требуется выполнить следующие пункты:

1. Подключение CSS-стилей и JavaScript-скрипта на html-странице.
2. Задание html-элемента с уникальным идентификатором:

```
<div id="map" class="map"></div>
```

3. Установление размеров созданного html–элемента с помощью CSS:

```
.map {
     height: 400px;
     width: 100%;
 }
```

4. Создание экземпляра класса `ol.Map`, который и инициализирует карту в html-элементе:

```
var map = new ol.Map(params)
```
(2)

Аргумент `params` в методе 2 — это объект различных параметров, каждую опцию которого следует рассмотреть отдельно:

- `target` — задает идентификатор элемента, созданного на шаге 2.
- `layers` — массив, который определяет список слоев, отображаемых на карте. Для создания тайлового слоя данное поле будет иметь вид:

```
layers: [
  new ol.layer.Tile({
    source: new ol.source.OSM()
  })
];
```

Операция `ol.layer.Tile()`создает слой, а параметр `source` — это

протокол, используемый для получения фрагментов карты. В данном примере используются тайлы с популярного сервиса OpenStreetMap [16].

- `view` — позволяет указывать центр карты, уровни масштабирования, проекцию и прочие настройки:

```
view: new ol.View({
  center: ol.proj.fromLonLat([37.41, 8.82]),
  zoom: 4
});
```

Поле `center` — географические координаты центра карты, которые с помощью метода `ol.proj.fromLonLat()` будут преобразованы в координаты проекции Меркатора (код EPSG:3857). Проекцию при необходимости можно изменить, передав ее код вторым аргументом в данный метод. Поле `zoom` — начальный уровень масштаба при инициализации карты.

Несомненно, в OpenLayers 4 количество операций для инициализации карты с начальным слоем, по сравнению с другими библиотеками, выше, однако эта особенность позволяет гибко настроить конфигурацию карты.

Слои данной библиотеки, добавленные на карту, формируют коллекцию подобно группам в Leaflet. Коллекция — это массив с дополнительными методами, которые позволяют манипулировать слоями, находящимися на карте. Для добавления слоя в коллекцию используется метод `addLayer(layer)`. По умолчанию слой отобразится поверх остальных слоев. Для того чтобы добавить слой в заданную позицию, во-первых, необходимо получить массив слоев с помощью метода `getLayers()`, а во-вторых, использовать метод `insertAt(index, layer)`, где `index` — позиция, в которую требуется вставить слой `layer`:

```
map.getLayers().insertAt(index, layer)
```

Для изменения конфигурации карты, например, ее проекции, можно вызвать метод `setView(view)`, где `view` — экземпляр класса `ol.View`, который был рассмотрен выше.

Настройка визуализации слоя осуществляется при помощи следующих методов:

- `setOpacity(opacity)` — устанавливает прозрачность слоя, где `opacity` — число из диапазона от 0 до 1;
- `setZIndex(zindex)` — устанавливает порядок отображения слоя на карте, где `zindex` — целое положительное число;
- `setVisible(visible)` — устанавливает видимость слоя, где параметр `visible` равен `true` или `false`;
- `setExtent(extent)` — устанавливает рамки видимости слоя, где `extent` — массив чисел вида `[minx, miny, maxx, maxy]`.

Данные методы вызываются для экземпляра класса `ol.layer.Layer`.

Для того что бы получить конкретный параметр слоя, в вышеописанных методах следует заменить префикс `set` на `get` и вызвать их без аргументов. Например, метод `getOpacity()` вернет значение прозрачности слоя.

Для удаления слоя используется метод `removeLayer(layer)`.

OpenLayers 4 поддерживает такие форматы пространственных данных, как WMS, WFS, векторные данные KML, GML, GeoJSON [3].

В большинстве случаев API данной библиотеки полностью покрывает требования проекта: она универсальна в использовании — может отображать как географические, так и не связанные с картами данные: маркеры, html-элементы, линии, полигоны, изображения, анимации, что помогает пользователю хорошо ориентироваться в приложении; имеет огромное сообщество, составляющее 175 разработчиков [17] и тысячи пользователей, которые помогают совершенствовать данный продукт [18] и предлагает большое количество обучающих материалов [19].

Подведение итогов анализа

В данном разделе были рассмотрены четыре наиболее распространенные и развитые библиотеки визуализации картографических данных: Google Maps, Polymaps, Leaflet и OpenLayers 4. Их сравнение проводилось на основе двенадцати критериев, выдвинутых в параграфе «Постановка задачи».

Результаты анализа по критерию «возможность управления коллекцией отображаемых слоев» вынесены в таблицу 1. В данной таблице, как уже было сказано выше, сложность операции измеряется как количество вызванных методов, необходимых для выполнения конкретного требования. Результаты анализа по остальным критериям приведены в таблице 2.

	OpenLayers 4	Polymaps	Leaflet	Google Maps
Сложность операции инициализации карты	3	3	2	1
Сложность операции добавления слоя с настройками визуализации по умолчанию	3	1	2	1
Сложность операции добавления слоя с пользовательскими настройками визуализации	4	1	3	2
Сложность операции изменения настроек визуализации слоя.	1	1	1	1
Сложность доступа к конкретному слою из отображаемого набора.	1	1	1	1
Сложность операции удаления слоя из карты	1	1	1	1

Таблица 1. Сравнение четырех библиотек отображения картографических данных по критерию «Возможность управления коллекцией отображаемых слоев» (единицы измерения — количество методов, вызванных программистом).

Примечания к таблице 1:

[1] Для настройки визуализации необходимо вызвать столько методов, сколько свойств требуется поменять у слоя (один вызов метода изменяет одно свойство).

	OpenLayers 4	Leaflet	Polymaps	Google Maps
Масштабирование	да	да	да	да
Навигация по карте	да	да	да	да
Размер библиотеки	465кб	154кб	32кб	[1]
Вывод дополнительной информации на область карты	да	да	да	да
Open Source	да	да	да	нет
Адаптированность под мобильные устройства	да	да	нет	да
Кросс-браузерность	да	да	[2]	да
Экспорт карт	да	да	нет	да
Поддерживаемые форматы пространственных данных	WMS, WFS, KML, GML, GeoJSON	WMS, GeoJSON, SVG, Canvas, KML[3]	GeoJSON, SVG	KML, GeoJSON, GeoRSS
Наличие материалов для изучения библиотеки	да	да	да	да
Геокодирование	да	[4]	нет	да
Проекции	EPSG:3857, EPSG:4326[5]	EPSG:3857, EPSG:3395, EPSG:4326, Simple	EPSG:3857	EPSG:3857

Таблица 2. Сравнение четырех библиотек отображения картографических данных по двенадцати критериям.

Примечания к таблице 2:

[1] Данное значение определить не удалось, поскольку библиотека не является свободно-распространяемой.

[2] Подходит для современных браузеров, поддерживающих язык разметки SVG.

[3] Поддержка KML формата осуществляется с помощью отдельного модуля, например, leaflet-omnivore.

[4] Поддержка осуществляется с помощью отдельного модуля, например, Leaflet GeoSearch.

[5] Для использования других официальных проекций требуется зарегистрировать их, используя соответствующие методы API библиотеки.

Заключение обзора

API рассмотренных выше библиотек понятен и хорошо задокументирован. Выполнение основных операций с картографическими данными схоже во всех библиотеках и не составляет труда с точки зрения JavaScript-разработчика.

Различие существует в возможностях, реализуемых библиотеками: количество доступных проекций без установки дополнительных модулей, разрешенные форматы пространственных данных, манипуляция слоями — стилизация, геокодирование, изменение порядка слоев и т. д.

Библиотеки Google Maps и Polymaps однозначно не подходят под требования O-GIS. Недостаток Google Maps заключается в том, что она не является библиотекой с открытым исходным кодом, привязана к единственной карте Google и использует единственную проекцию. Polymaps, в свою очередь, отображает данные только в векторных форматах.

Leaflet — достойный конкурент, но данная библиотека нацелена на быструю скорость работы с картами в ущерб опциональности API. Так LeafLet поддерживает только WMS протокол передачи картографических данных, в то время как OpenLayers 4 может работать с абсолютно любым источником, поддерживающим OGC стандарт. На официальном сайте OpenLayers 4 расположено 162 примера работы с данной библиотекой [20] — у LeafLet их всего 14 [21]. Для OpenLayers 4 кроме документации API существует комплексный обзор библиотеки, который находится на GitBook [22]. К тому же, для использования проекции, отсутствующей в Leaflet, придется подключить дополнительный модуль и потратить время на его изучение. В OpenLayers 4 это сделать намного проще, зарегистрировав нужную проекцию, используя при этом всего один метод.

Дальнейшая модификация O-GIS будет осуществляться с помощью библиотеки отображения картографических данных OpenLayers 4, так как

именно эта библиотека предоставляет широкую опциональность API и гибкость разработки.

Глава 1. Архитектура текущей версии O-GIS

В данной главе будет представлена архитектура системы O-GIS до этапа модификации.

Архитектура приложения O-GIS была спроектирована с учетом «тонкого» клиента. Это значит, что бóльшая часть задач по обработке информации выполняется на сервере, а клиент лишь отображает запрошенные данные и выполняет роль интерфейса для ввода различных команд. Такое решение было принято в связи с тем, что конечный продукт O-GIS ориентирован, прежде всего, на мобильные устройства, мощность которых мала по сравнению с современными компьютерами, поэтому объемные вычислительные операции, такие как растровая реклассификация и растровая алгебра, выполняются на сервере.

На стороне сервера данная ГИС имеет следующее программное обеспечение: web-сервер приложения Apache 2 [23], картографический web-сервер GeoServer [24], база данных PostGIS [25], фреймворк Symfony [26], написанный на языке программирования PHP.

На стороне клиента для отображения картографических данных используется JavaScript библиотека OpenLayers 2 [1]. Для построения пользовательского интерфейса — библиотека jQuery [27].

Далее компоненты системы O-GIS будут рассмотрены более подробно.

1.1. Серверная часть

1.1.1. PostGIS

PostGIS — это надстройка над объектно-реляционной СУБД PostgreSQL, которая добавляет поддержку географических объектов. Основным достоинством данного расширения является использование языка SQL, который дополняется операторами и функциями для работы с пространственными данными. PostGIS позволяет обрабатывать растровые и векторные

форматы данных, добавляет функции объединения, пересечения географических объектов, реклассификации растра, растровой алгебры, вычисления значений длин, площадей конкретной области пространственного объекта.

Как известно, географические данные занимают большие объемы памяти, поэтому данная СУБД предоставляет собственные инструменты для загрузки, выгрузки данных, а также для их представления. Для PostGIS разрабатываются программы и сторонними разработчиками, организациями. Большинство из них находится в открытом доступе. Например, программное обеспечение GeoServer, которое используется в системе O-GIS и описывается в следующем разделе.

Для системы O-GIS программный модуль PostGIS используется как для хранения геопространственных данных, так и информации, связанной с пользователями и созданными ими проектами.

1.1.2. GeoServer

Картографический сервер, написанный на языке Java, который позволяет пользователям просматривать и редактировать пространственные данные. GeoServer использует открытые стандарты, установленные международным консорциумом Open Geospatial Consortium (OGC), что дает гибкость при создании карт и при их совместном использовании.

Для передачи картографических данных по сети такой сервер использует стандарт WMS, предоставляя простой http-интерфейс для реализации запросов карт из одной или нескольких распределенных баз данных. Так, для получения конкретной карты на сервер GeoServer необходимо передать идентификатор карты, желаемые проекцию и формат ответа. В свою очередь, GeoServer запросит информацию из базы данных, отобразит ее в формат, указанный в запросе, и отправит в ответ.

1.1.3. Symfony

Бесплатный фреймворк, написанный на языке программирования PHP. С его помощью можно быстро разрабатывать серверную часть web-приложений.

Symfony использует концепцию Model-View-Controller. Компонент Model — это данные приложения и методы работы с ними. View отображает данные Model пользователю и реагирует на их изменение. Controller обеспечивает связь между пользователем и системой — изменяет данные Model в ответ на действия пользователя. Достоинство описанного подхода заключается в отделении данных, бизнес-логики, связанной с этими данными, от их отображения — компоненты View и Controller можно изменять, не затрагивая компонент Model; нет необходимости переиспользовать код приложения — с одной моделью можно связывать несколько представлений (которые будут отображать данные по-разному) и менять контроллеры, реакция на действия пользователя которых будет различна [28].

Symfony с помощью дополнительных инструментов поддерживает ORM технологию, которая представляет данные, что хранятся в реляционной базе данных, в виде объектов — абстракций, описывающих конкретную сущность предметной области, которые используются внутри приложения. Такой подход снимает некоторые обязанности с разработчика и снижает вероятность появления ошибок в модуле взаимодействия приложения с базой данных: нет необходимости вручную собирать информацию по всем таблицам базы данных и инкапсулировать ее в объектах [29].

В проекте O-GIS фреймворк Symfony является логическим ядром системы — он отправляет отображения браузеру в виде html-страниц, обрабатывает запросы пользователя, взаимодействует с картографическим сервером GeoServer и базой данных PostGIS для обработки пространственных данных и информации, связанной с пользователями.

Взаимодействие описанных выше компонентов серверной части системы O-GIS изображено на рисунке 1. Функционирование O-GIS можно представить в следующем виде:

- пользователь запрашивает информацию, не связанную с геопространственными данными: его сообщения, заметки, список участников системы, проектов и т. д., либо метаинформацию, которая описывает пространственные данные, но не является их частью: список слоев, композиций, палитр;
- пользователь запрашивает отображение слоев, композиций, палитр в браузере, координаты конкретной точки слоя.

Оба варианта обрабатывает фреймворк Symfony. В первом варианте Symfony запрашивает требуемую информацию у базы данных PostGIS и возвращает пользователю в обработанном виде. Во втором варианте Symfony передает запрос картографическому серверу GeoServer, выполняя промежуточную роль (прокси); GeoServer запрашивает пространственные данные у базы данных PostGIS, обрабатывает их и передает обратно серверу приложений, который, в свою очередь, перенаправляет их клиенту. Клиент отображает принятые данные в браузере. Такой способ передачи данных избавляет от кросс-доменных запросов со стороны клиента, которые, как известно, запрещены в целях безопасности.

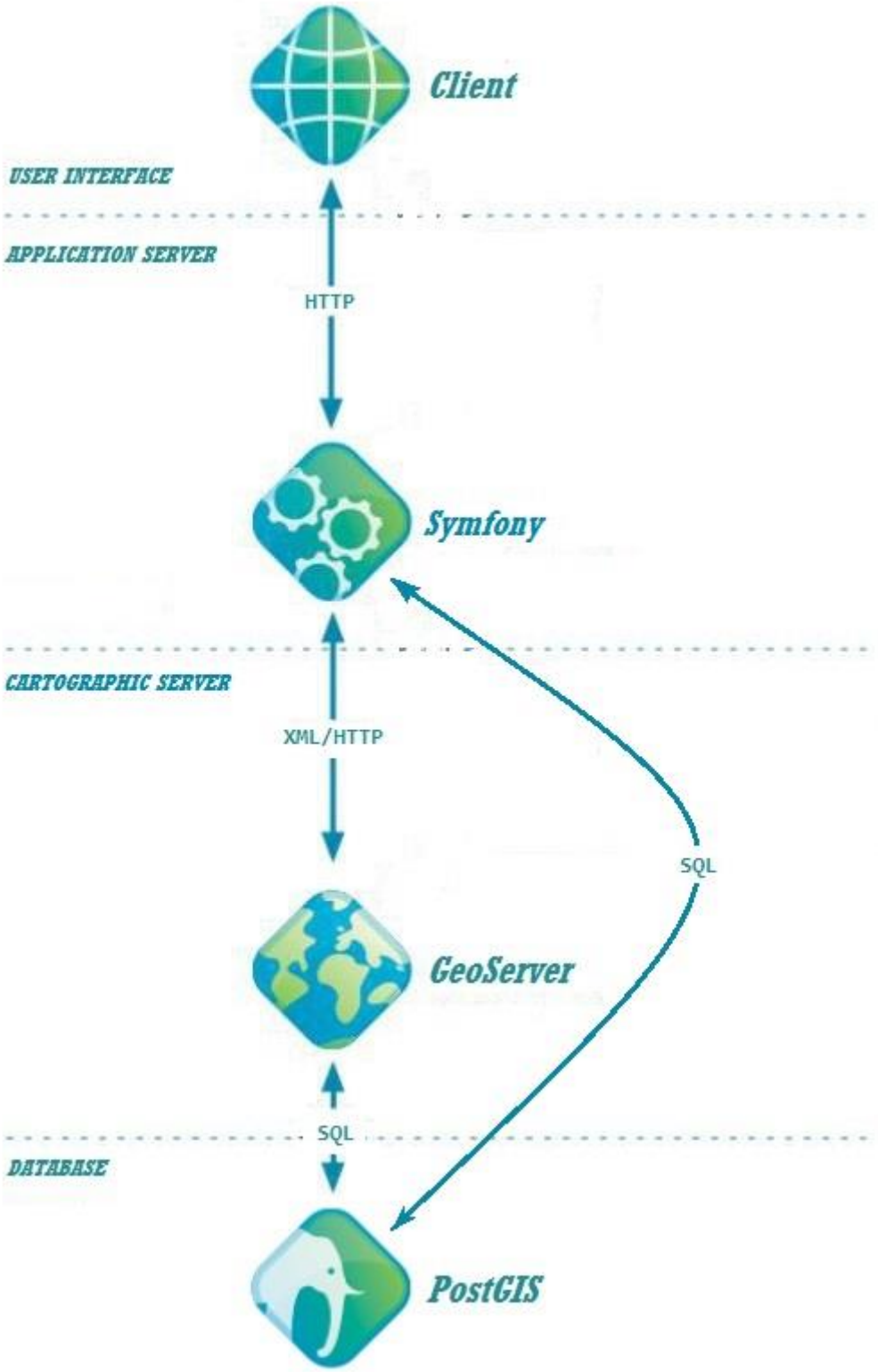

Рис. 1. Схема взаимодействия компонентов системы O-GIS.

1.2. Клиентская часть

1.2.1. OpenLayers 2

Библиотека визуализации картографических данных на стороне браузера, которая находится в бесплатном доступе с открытым исходным кодом. С ее помощью можно отображать растровые и векторные данные, маркеры, html-элементы, линии, полигоны, изображения, анимации. OpenLayers 2 поддерживает стандарт OGC, имеет большое количество обучающих материалов и огромное сообщество, которое помогает улучшать данный продукт.

Как уже было сказано выше, вторая версия библиотеки является устаревшей. На момент написания данной работы существует четвертая версия, которая была описана в главе «Обзор библиотек визуализации картографических данных».

В проекте O-GIS библиотека OpenLayers 2 для передачи геопространственных данных по сети использует протокол WMS, который позволяет передавать части карты конкретного уровня масштабирования параллельно.

1.2.2. jQuery

Быстрая небольшая и многофункциональная библиотека JavaScript. Предоставляет средства для манипулирования элементами DOM-дерева, управления событиями браузера, визуализации анимаций; упрощает использование асинхронных запросов к удаленным узлам. Некоторые конструкции данной библиотеки реализуют уже существующие функции JavaScript, однако они заметно короче и обеспечивают гибкую настройку. Вес jQuery в минифицированном виде занимает всего 88 Кб.

Библиотека jQuery имеет дополнительные модули, которые расширяют ее функциональность. В проекте O-GIS помимо самой jQuery используется плагины jQuery UI и jsTree. jQuery UI позволяет быстро создавать пользовательские интерфейсы, затратив минимальные усилия на плани-

рование дизайна окон и их элементов. С помощью плагина jsTree легко генерировать интерактивные деревья, которые отображают элементы файловой системы — иерархию вложенных папок и файлов. В O-GIS модуль jsTree используется для представления каталогов пользователей, в которых хранятся слои, композиции и палитры.

Библиотека jQuery больше подходит для малых или средних проектов, поскольку в больших проектах, таких как O-GIS, если не следовать хорошим практикам, код быстро разрастается, данные смешиваются с их представлением, программные части дублируются, в итоге отлаживать приложение становится все сложнее, ошибки возникают все чаще. В связи с этим автором данной работы было предложено заменить данный модуль системы на библиотеку ReactJS, которая будет описана в следующей главе.

Глава 2. O-GIS для работы на мобильных устройствах

В данной главе будут подробно описаны основные инструменты разработки web-приложений, которые применялись для модернизации системы O-GIS, для того чтобы с последней было возможно работать на мобильных устройствах.

2.1. Клиентская часть

Для реализации поставленной задачи данной ВКР основной акцент был уделен клиентской части системы, поскольку главная цель разработки приложений под мобильные устройства состоит в построении качественного, адаптивного пользовательского интерфейса на стороне клиента для отображения в браузере.

Как уже было сказано в главе «Обзор библиотек визуализации картографических данных» для визуализации пространственных данных была выбрана библиотека OpenLayers 4. Остальные инструменты разработки на клиентской стороне применяются, во-первых, потому что автор данной работы имел некоторый опыт использования их в реализации web-приложений, а во-вторых, они подошли для решения поставленной задачи и, более того, полностью покрывают ее требования. Эти инструменты будут разобраны далее.

2.1.1. ESlint

ESlint — инструмент, предназначенный для статического анализа JavaScript кода в реальном времени [30]. Статический — означает, что программа в данный момент не выполняется, а анализ ее ведется в фоновом процессе.

С помощью инструмента ESlint стиль написания кода во всех проектах будет одинаковым, поскольку он генерирует предупреждения, если какая-либо конструкция языка имеет несогласованный с правилами синтаксис, да-

же если последняя написана верно. Данное преимущество относится и к командным проектам — разработчики могут быстро ориентироваться в чужом коде. Второе, не менее важное достоинство — уведомление программиста об ошибках синтаксиса кода. Последний быстро сориентируется и найдет ошибку в приложении, на которую укажет данный инструмент — разработчику не придется компилировать программу, что бы отладить код проекта.

ESlint — не единственный модуль статического анализа JavaScript кода, который доступен в открытом доступе. Одним из первых таких инструментов был JSLint, разработанный американским программистом Дугласом Крокфордом, который известен созданием текстового формата JSON, что используется для передачи данных по сети в web-приложениях [31, 32]. Главный недостаток JSLint заключается в том, что его нельзя настроить под свои требования, указать свои правила семантики и синтаксиса кода, т. е. отсутствует гибкость, которая так нужна при разработке современных web-приложений.

Главным приемником JSLint стал инструмент статического анализа JSHint. JSHint — это модернизированный проект JSLint, который позволяет настраивать правила анализа для каждого разработчика индивидуально [33, 34].

ESlint, в свою очередь, отличается от инструментов JSLint и JSHint тем, что он анализирует не текст программ, а сгенерированное в памяти абстрактное синтаксическое дерево. Внутренними узлами такого дерева являются операторы языка программирования, а листья — операнды, т. е. переменные и константы. Это добавляет возможность написания своих плагинов, которые генерируют новые правила для анализа кода. ESlint позволяет гибко настроить правила анализа под требования команды. Для распознавания специфичного синтаксиса, например JSX — смесь JavaScript и XML, существуют плагины, которые легко установить и использовать.

2.1.2. Babel

JavaScript зародился в 1995 году в корпорации Netscape Communications. Изначально его официальное название было LiveScript, однако вследствие популяризации на тот момент языка программирования Java, LiveScript переименовали в JavaScript — как ни странно, общего у него с языком Java мало. Далее JavaScript необходимо было стандартизировать — формализовать синтаксис и семантику. Для этого разработали стандарт с идентификационным номером ECMA-262. ECMA — это ассоциация стандартизации информационных и коммуникационных систем, которая была создана в 1961 году.

На сегодняшний день существуют несколько стандартов JavaScript. В современных браузерах используется ECMA5, однако это не последняя спецификация — существует еще 6-ая (именуемая ECMA2015), которая будет поддерживаться в браузерах в будущем, и 7-ая (ECMA2016), которая на данный момент разрабатывается [35]. Для реализации web-приложений, написанных по последнему стандарту ECMA, был разработан программный модуль Babel.

Babel — это инструмент, компилирующий JavaScript в стандарт, который поддерживается в среде разработки — например, в современных браузерах. Данный инструмент позволяет разработчикам web-приложений писать свои программы, используя последние возможности языка. Его применяют такие крупные компании, как Facebook, Google, Yahoo!. На текущий момент Babel стал незаменимым инструментом JavaScript разработчика [36].

По умолчанию Babel оставляет код приложения прежним. Для трансформации кода необходимо подключить плагины, каждый из которых определяет трансформацию отдельной конструкции языка. Возможно подключить готовый набор плагинов, называемых пресетами. Например, пресет `es2015` будет трансформировать код стандарта ECMA2015 в стандарт ECMA5.

Для инструмента Babel существуют экспериментальные пресеты, которые трансформируют конструкции языка, не вошедшие ни в один из стандартов. Такие конструкции могут и позже не войти в следующие спецификации, поэтому использовать их нужно с осторожностью.

Комитет TC39, который разрабатывает стандарты JavaScript, утвердил следующие этапы реализации ECMA:

- Stage 0 — идея введения нового синтаксиса;
- Stage 1 — конкретные предложения введения нового синтаксиса;
- Stage 2 — черновой вариант спецификации;
- Stage 3 — полная спецификация и начальная реализация поддержки браузеров;
- Stage 4 — добавление спецификации в следующий выпуск стандарта.

Названия экспериментальных пресетов Babel соответствуют названиям вышеперечисленных этапов. Пресеты `stage-0`, `stage-1` и `stage-2`, могут изменить свое поведение в будущем, тогда как `stage-3` менее подвержен изменениям, а синтаксические конструкции пресета `stage-4` будут использоваться в следующем стандарте.

2.1.3. Webpack

Современная идеология разработки программных продуктов на языке JavaScript предполагает наличие некоторых свойств: модульность — разделение логики в отдельные файлы проекта, минификация — объем занимаемой web-приложением памяти должен быть минимально возможным для быстрой загрузки в браузерах, и использование препроцессора — надстройка над программным модулем, которая добавляет дополнительные возможности с помощью новых синтаксических конструкций. Однако при разработке про-

ектов, используя только JavaScript, CSS и HTML, данные свойства воплотить сложно. Для их реализации существуют сборщики проектов.

Webpack — это программный модуль JavaScript, одна из главных функций которого — сборка проекта. При обработке приложения, он рекурсивно создает граф зависимостей, который включает в себя все модули, необходимые для работы приложения, а затем упаковывает их в небольшое количество пакетов (часто — только в один) для быстрой загрузки браузером. Все, что необходимо для выполнения таких действий — указать при настройке Webpack точку входа в приложение (начальный файл, который ссылается на остальные зависимости) и точку выхода — местоположение для скомпилированного конечного файла [37].

По умолчанию Webpack «понимает» только JavaScript файлы. Для обработки CSS, HTML файлов, изображений и прочих зависимостей используются загрузчики. Например, существует загрузчик для компиляции препроцессоров CSS в CSS формат. Также имеется загрузчик Babel — при его добавлении разрабатываемое приложение будет работать на всех современных браузерах.

Еще одним мощным инструментом сборщика Webpack является набор плагинов. Плагин позволяет изменять логику сборки, выполняя специальные функции над частями проекта. Например, существуют плагины для минификации кода, удаления дублирующего кода, сборки CSS таблиц в один файл и тому подобное.

Дополнительным инструментом Webpack является модуль HMR (Hot Module Replacement). Он позволяет компилировать измененные модули приложения в реальном времени без перезагрузки страницы браузера и пересборки проекта.

2.1.4. React

Открытая мощная быстрая JavaScript библиотека, разработанная компанией Facebook. Предназначена для построения адаптивных и интерактивных пользовательских интерфейсов. Основная концепция данного инструмента заключается в использовании JSX синтаксиса — смесь языковых конструкций JavaScript и разметки XML. JSX позволяет строить клиентскую часть приложения с помощью декларативного подхода — разработчик описывает структуру модулей программы, но не указывает каким образом данные модули отображаются, перерисовываются в DOM-дереве браузера, удаляются из него — такие функции выполняет React, затратив минимум ресурсов и времени [38].

Поскольку синтаксис JSX не поддерживается в браузерах, React накладывает дополнительное требование к разрабатываемому проекту — подключение инструмента, который будет трансформировать JSX в языковые конструкции JavaScript. С такой задачей хорошо справляется программный модуль Babel, описанный выше. React также позволяет отказаться от JSX синтаксиса и разрабатывать приложения на нативном JavaScript, однако программный код при таком подходе будет более громоздким и менее понятным.

Второй особенностью данной библиотеки является Virtual DOM — дерево, которое построено в памяти приложения React и содержит динамическую структуру html-документа. Оно необходимо для оптимизации операции отображения элементов в DOM-дереве. Жизненный цикл приложения React можно описать с помощью следующих этапов:

- Virtual DOM инициализируется при первом рендеринге приложения в браузере и полностью клонируется в DOM;
- повторная генерация Virtual DOM возникает в ответ на действия пользователя;

- Virtual DOM сравнивается с DOM’ом браузера и, если элементы двух деревьев рознятся — React перерисовывает только измененные элементы.

Как известно, операции, выполняемые над DOM-деревом браузера, являются дорогостоящими, а благодаря концепции Virtual DOM количество обновлений DOM-дерева стремится к минимальному.

Библиотека React предполагает разделение клиентской части приложения на компоненты — строительные блоки программы, каждый из которых имеет свою функциональность, логическую структуру и связан, чаще всего, с единственным элементом пользовательского интерфейса. Такие компоненты можно легко изменять, не затронув остальные блоки программы, переносить в другой проект и вкладывать друг в друга. Главное достоинство такого подхода — масштабируемость приложения.

Еще одним плюсом данного инструмента является возможность разработки мобильных приложений на JavaScript с помощью модуля React Native для платформ IOS и Android. Таким образом, разработчики web-приложений с легкостью могут создавать программные продукты, которые запускаются вне браузера [39].

Для реализации поставленной в данной ВКР задачи было принято решение заменить в клиентской части системы O-GIS библиотеку jQuery на инструмент React. С его помощью будут построены пользовательские адаптивные и интерактивные интерфейсы, доступные как на стационарных компьютерах, так и на мобильных устройствах.

2.1.5. Create React App

Разработка современных React приложений предполагает наличие по крайней мере следующих инструментов: статический анализатор кода, трансформатор неподдерживаемого на данный момент JavaScript синтаксиса, сборщик проекта. Данные возможности реализуют программные модули

ESlint, Babel и Webpack соответственно, которые были описаны выше. Для того чтобы использовать их в своем проекте, необходимо написать различные конфигурационные файлы, которые будут определять поведение этих инструментов. Однако это отнюдь не тривиальная задача — нужно четко понимать значение каждой конфигурации и уметь связывать их между собой — правила могут противоречить друг другу. Также обновление какого-либо инструмента может потребовать изменение конфигурации.

Команда разработчиков React создала библиотеку Create React App, которая решает вышеуказанные проблемы. Create React App позволяет реализовывать React приложения с нулевой конфигурацией — все настройки будут вшиты внутрь данного модуля и скрыты от программиста. Разработчик может сразу же приступить к написанию логики приложения — ему не придется его настраивать. Управление проектом осуществляется с помощью трех команд — сборка проекта, запуск приложения в режиме разработки и запуск тестов [40].

Поскольку конфигурация проекта скрыта от программиста, это накладывает определенные ограничения: например, в приложении нельзя использовать некоторые экспериментальные конструкции JavaScript (нулевой этап разработки стандарта JavaScript). Для добавления своих правил в конфигурацию системы либо для их изменения можно «развернуть» проект так, чтобы все настройки были доступны. Однако использовать такой подход нужно в крайнем случае, поскольку данная команда необратима, а дальнейшее обновление проекта будет невозможно.

В режиме разработки Create React App запускает свой сервер, который, во-первых, будет отдавать все статические файлы браузеру, а во-вторых, будет автоматически обновлять страницу приложения при изменении модулей проекта — это значительно упрощает и ускоряет процесс разработки.

Одной из важных возможностей, которую предоставляет данный инструмент, является интеграция в приложение с существующим сервером. В

таком случае последовательность действий в режиме разработки будет следующей:

- при первом посещении приложения сервер Create React App пришлет статические файлы;
- последующие запросы будут перенаправлены серверу приложения, в которое интегрирован Create React App (Create React App в режиме прокси).

Данная особенность позволяет использовать инструмент Create React App в проекте O-GIS, в котором, как известно, сервер приложения уже реализован. Схема взаимодействия описанных частей изображена на рисунке 2.

Рис. 2. Схема взаимодействия модуля Create React App с системой O-GIS в режиме разработки.

2.1.6. Material-UI

Библиотека для построения пользовательского интерфейса, реализующая концепции Material Design. Material Design — спецификация, представляющая классические принципы адаптивного дизайна мобильных и браузерных приложений, разработанная компанией Google. Подходы Material Design позволяют создавать консистентные унифицированные визуально привлекательные интерфейсы — пользователю будет удобно выполнять свою работу, он будет быстро ориентироваться в приложении на любом устройстве независимо от размера экрана [41].

Material-UI предоставляет разработчику множество различных компонент, написанных на React, из которых можно строить адаптивный и интерактивный пользовательский интерфейс: кнопки, формы, диалоговые окна,

таблицы и многое другое. Задача разработчика — продумать логику клиентской части и связать такие React-компоненты между собой. Очевидно, что такой подход позволяет больше времени уделить бизнес логике приложения [42].

С помощью библиотеки Material-UI будут реализованы пользовательские интерфейсы в модифицируемой системе O-GIS.

2.2. Серверная часть

Серверная часть системы O-GIS будет подвержена меньшим изменениям в сравнении с клиентской стороной проекта. Архитектура приложения, представленная на рисунке 1, в модифицированной ГИС останется прежней: связка фреймворка Symfony, картографического web-сервера GeoServer и базы данных PostGIS. Однако для решения задачи данной ВКР требуется изменить часть логического ядра системы, реализованного с помощью Symfony. Необходимые изменения будут подробно описаны далее.

2.2.1. Передача запросов посредством REST

В системе O-GIS до этапа модификации использовался подход рендеринга на стороне сервера. Это означает, что при каждом запросе сервер обрабатывал html-шаблон, внедрял в него необходимые данные (например, данные о пользователе), генерировал html-страницу и отдавал ее браузеру. Минус такого подхода заключается в генерации многочисленных запросов на сервер для отображения различных страниц либо диалоговых окон этих страниц в браузере.

Использование библиотеки React в клиентской части модифицируемой ГИС подразумевает рендеринг на стороне клиента — сервер отдает браузеру пустую html-страницу, содержимое которой заполняется с помощью JavaScript. Преимущество такого подхода — снижение нагрузки на сервер, поскольку, во-первых, серверу не нужно выполнять операцию шаблонизации, а во-вторых, загрузка пользовательского интерфейса выполняется только один

раз — все React-компоненты в виде JavaScript функций будут подгружены на клиент после первого запроса на сервер, последующие запросы — получение определенных пользовательских либо геопространственных данных, а не html-страниц. Недостаток подхода — требуется больше времени для первоначального отображения html-страницы в браузере.

Поскольку в модифицируемой системе O-GIS тип рендеринга изменится, данные, которые ранее внедрялись в html-шаблон, решено было получать посредством REST. REST — это стиль архитектуры приложения, описывающий взаимодействие его компонент — клиента и сервера. Парадигма REST проста в использовании и не накладывает ограничений на протокол передачи данных, на формат данных. При таком подходе запрос на получение информации с сервера определяется глобальным идентификатором URL и должен быть составлен так, чтобы компонент системы получил всю необходимую информацию для его обработки — сервер не хранит состояние клиента, а получает его в теле запроса [43].

2.2.2. Определение типа устройства

Конечная ГИС должна хорошо функционировать для всех типов устройств: мобильных, планшетных или десктопных. Для десктопного типа система O-GIS работает в полной мере безотказно — все предоставляемые пользователю функции, такие как композиция геопространственных данных, растровая реклассификация, растровая алгебра, изменение порядка слоев и т. п., выполняются без ошибок, а интерфейс удобен и интерактивен. Для мобильных устройств (в эту категорию можно отнести и планшетный тип) O-GIS должна выполнять безотказную работу после этапа модификации. В связи с вышесказанным на сервере ГИС требуется определять тип устройства, и в зависимости от последнего отдавать десктопную (до этапа модификации) или мобильную (после этапа модификации) версию O-GIS.

Распознать тип устройства можно с помощью User-Agent — строки заголовка HTTP запроса, которая содержит информацию о клиенте: название и версию операционной системы, ее разрядность, название и версию браузера, язык и прочее. В частности, такая строка включает информацию о типе устройства, с которого зашел пользователь [44].

Схема функционирования ГИС в зависимости от значения User-Agent изображена на рисунке 3.

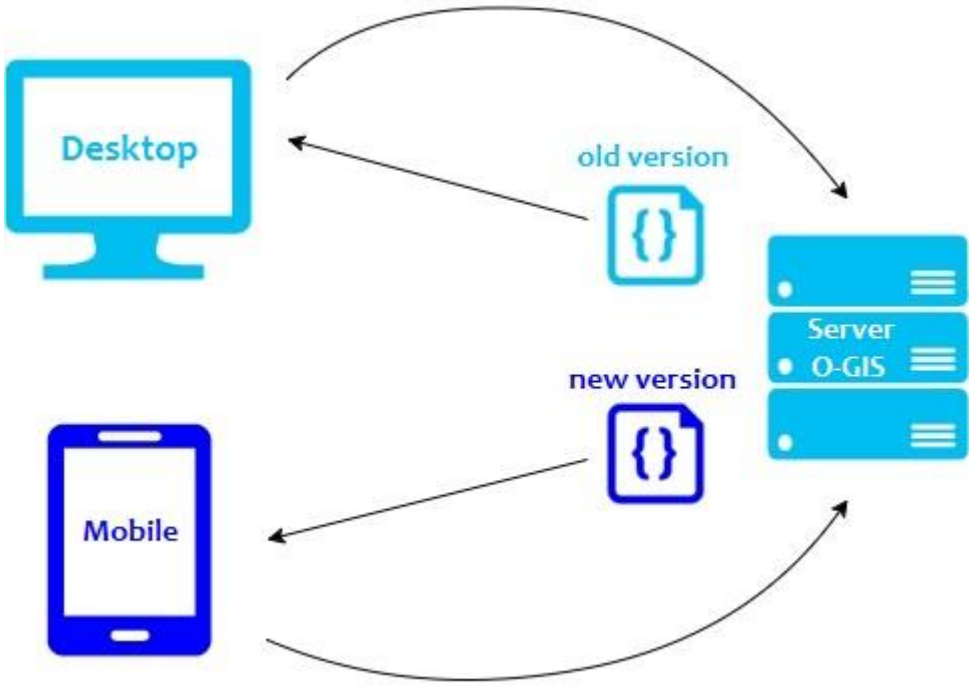

Рис. 3. Схема функционирования модифицированной системы O-GIS в зависимости от типа устройства пользователя.

Глава 3. Полученные результаты

В результате выполненной работы был модифицирован модуль открытой многопользовательской географической информационной системы O-GIS, отвечающий за визуализацию геопространственных данных. В качестве инструментов для модификации клиентской части модуля использовались статический анализатор кода ESlint, библиотека React, программный модуль Create React App, библиотека Material-UI. Проект O-GIS был переписан на новую четвертую версию библиотеки визуализации картографических данных OpenLayers. В серверной части системы O-GIS была изменена часть логического ядра, функционирующего с помощью фреймворка Symfony: данные, которые внедрялись в html-страницу, отображавшую модуль O-GIS в браузере, до этапа модификации, теперь передаются посредством REST парадигмы; в зависимости от типа устройства, с которого зашел пользователь, отсылается предыдущая (в случае десктопного типа) или модифицированная (в случае мобильного типа) версия модуля O-GIS.

Модифицированная часть системы O-GIS удовлетворяет требованиям, поставленным в разделе «Постановка задачи», как на мобильных, так и десктопных устройствах, а именно:

- Элементы пользовательского интерфейса:
    - Адаптивны — размеры элементов меняются в зависимости от размеров экрана браузера.
    - Интерактивны — незамедлительно реагируют на действия пользователя.
    - Одинаково функционируют как на кнопочных, так и на сенсорных устройствах.
    - Асинхронны — выполняют соответствующую операцию без перезагрузки страницы.

    - Атомарны — выполняют единственное бизнес требование.
- Окно, содержащее элементы управления редактором:
    - Сворачивается по умолчанию.
    - Осуществляет функцию изменения порядка слоев.
- Редактор композиций слоев предоставляет функции масштабирования, навигации, геокодирования, стилизации, растровой реклассификации, растровой алгебры слоев, которые теперь удобно выполнять на мобильных устройствах.

Следующие требования возможно будет проверить после запуска модифицированного модуля в рабочем режиме, опросив пользователей системы:

- Оптимальные размеры элементов на любом типе устройства.
- Оптимальное положение элементов на экране.
- Понятный пользователю внешний вид элементов, отражающий специфику выполняемой операции.

3.1. Визуальное сравнение редакторов

В текущем параграфе будет представлен модифицированный редактор композиций O-GIS в сравнении с его предыдущей версией.

На рисунках 4, 6, 8, 10, 12 и 5, 7, 9, 11, 13 изображены скриншоты редактора композиций системы O-GIS на экране мобильного устройства размером 568 на 320 пикселей до и после этапа модификации соответственно. На скриншотах модуля проекта O-GIS после этапа модификации отчетливо прослеживается его адаптированность под мобильные устройства, особенно в сравнении с предыдущей версией редактора.

Рис. 4. Скриншот редактора композиций системы O-GIS со свернутой панелью управления до этапа модификации (экран размером 568 на 320 пикселей).

Рис. 5. Скриншот редактора композиций системы O-GIS со свернутой панелью управления после этапа модификации (экран размером 568 на 320 пикселей).

Рис. 6. Скриншот редактора композиций системы O-GIS с развернутой панелью управления до этапа модификации (экран размером 568 на 320 пикселей).

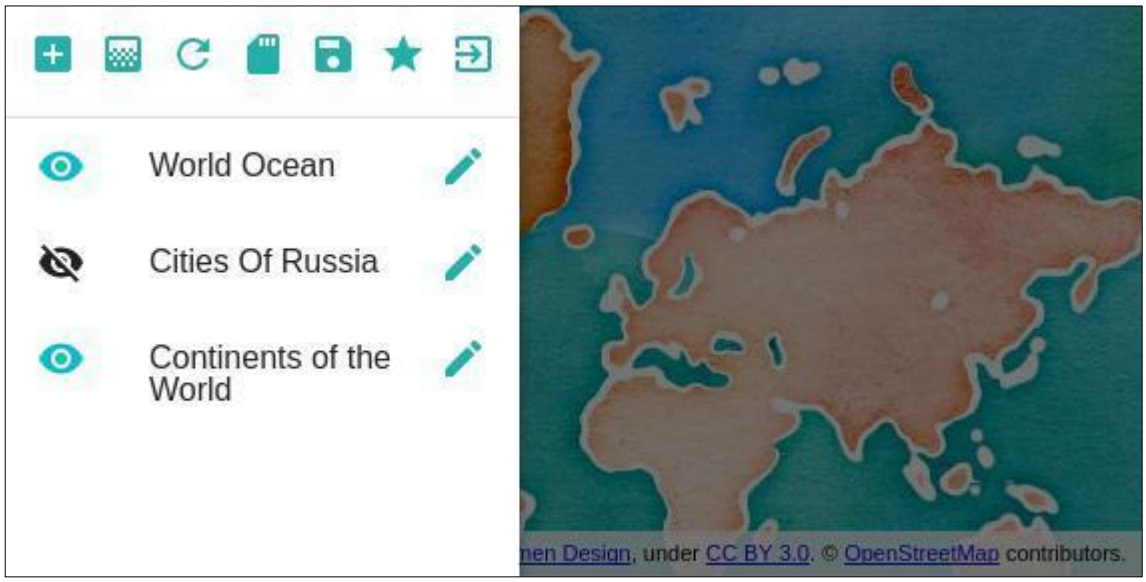

Рис. 7. Скриншот редактора композиций системы O-GIS с развернутой панелью управления после этапа модификации (экран размером 568 на 320 пикселей).

Рис. 8. Скриншот редактора композиций системы O-GIS с развернутой панелью управления до этапа модификации, мобильное устройство находится в вертикальном положении (экран размером 568 на 320 пикселей).

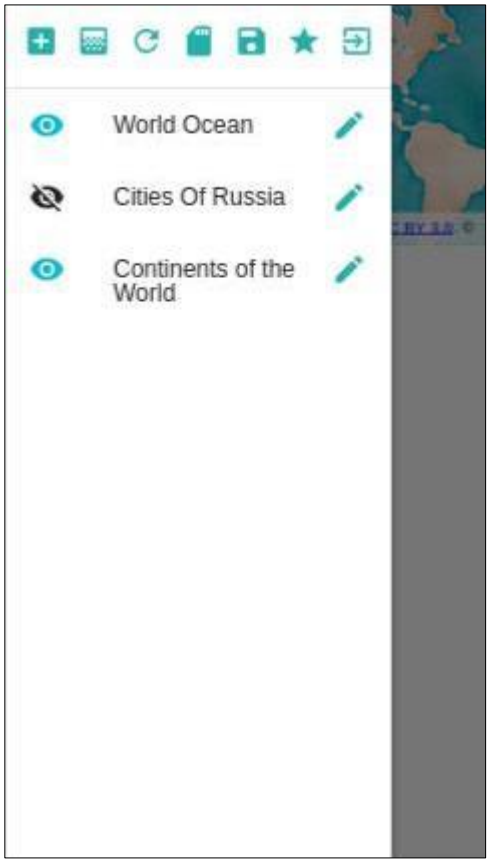

Рис. 9. Скриншот редактора композиций системы O-GIS с развернутой панелью управления после этапа модификации, мобильное устройство находится в вертикальном положении (экран размером 568 на 320 пикселей).

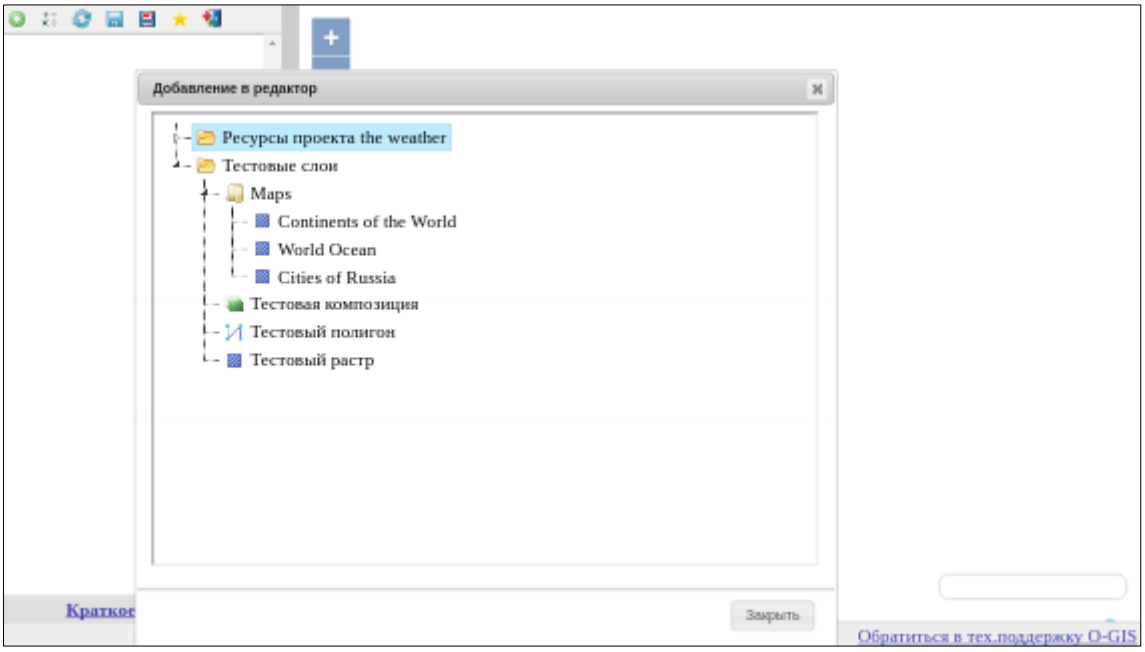

Рис. 10. Скриншот диалогового окна «Добавление геоданных в редактор» редактора композиций системы O-GIS до этапа модификации (экран размером 568 на 320 пикселей).

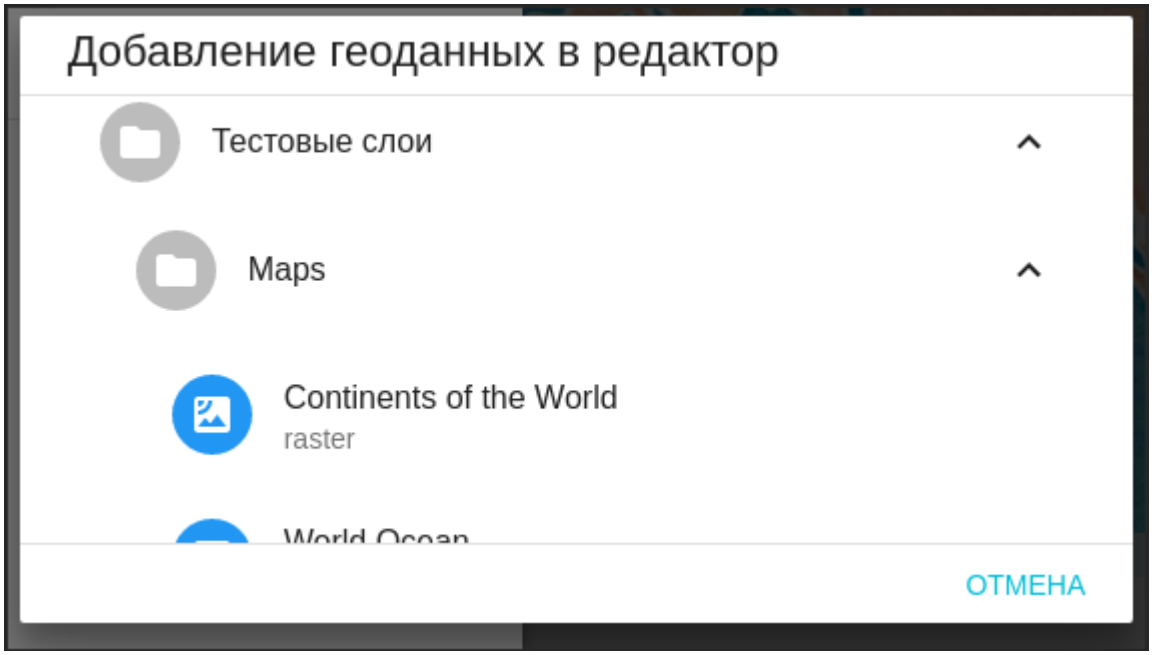

Рис. 11. Скриншот диалогового окна «Добавление геоданных в редактор» редактора композиций системы O-GIS после этапа модификации (экран размером 568 на 320 пикселей).

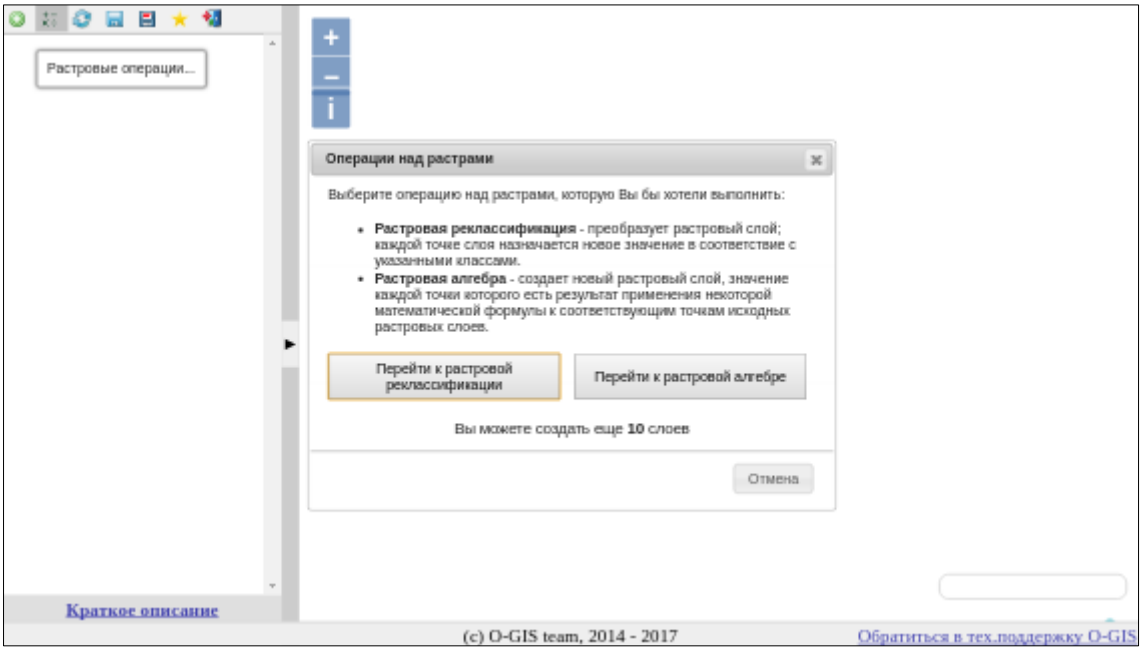

Рис. 12. Скриншот диалогового окна «Растровые операции» редактора композиций системы O-GIS до этапа модификации (экран размером 568 на 320 пикселей).

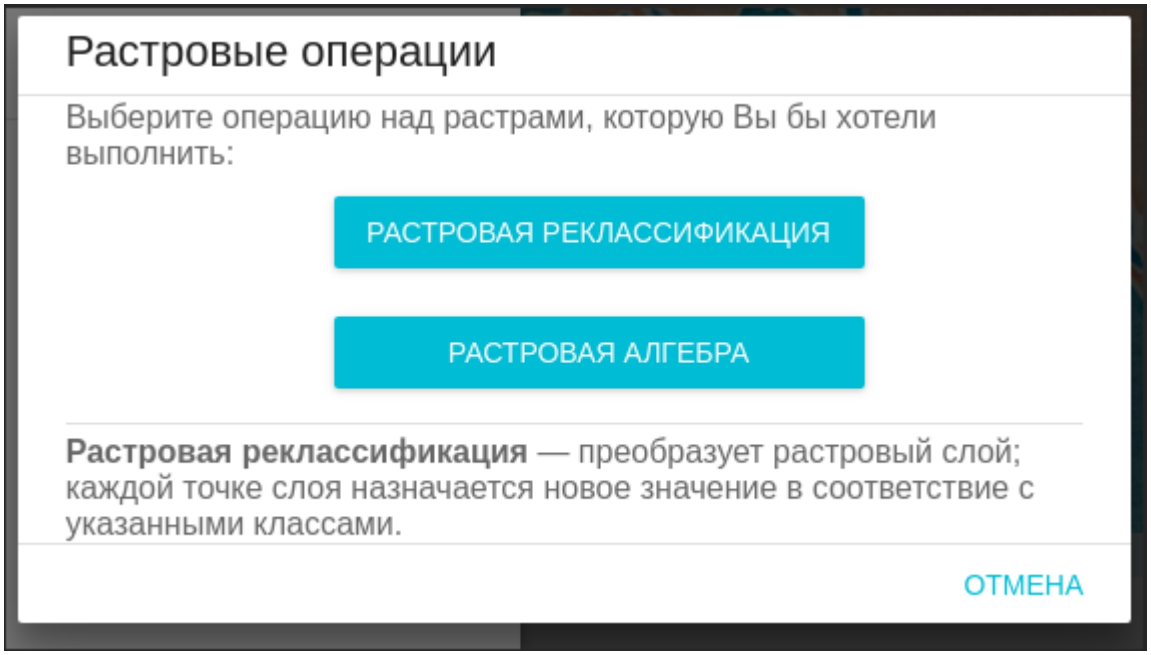

Рис. 13. Скриншот диалогового окна «Растровые операции» редактора композиций системы O-GIS после этапа модификации (экран размером 568 на 320 пикселей).

Выводы

В данной работе был модифицирован редактор композиций геопространственных данных открытой многопользовательской географической информационной системы O-GIS, а именно:

Были решены проблемы запуска системы на локальной машине на начальном этапе разработки, которые связаны с устаревшими дистрибутивами, используемыми в системе. Были изучены субъекты системы (потенциальные пользователи), объекты системы (информация о пользователях, геопространственные данные), бизнес-процессы, возникающие между субъектами и объектами. Была исследована структура системы: взаимосвязь различных модулей, их логическая составляющая. Было произведено знакомство с инструментами, на которых базируется данная ГИС. Были выдвинуты требования к конечному решению поставленной задачи. Был спроектирован план работ по модификации системы, разделенный на три этапа:

I. Выбор библиотеки визуализации картографических данных, подходящей под требования O-GIS, и дальнейшее переписывание проекта на новую библиотеку.
II. Модификация клиентской части модуля O-GIS.
III. Модификация серверной части модуля O-GIS.

Автором данной ВКР был предложен стек технологий, обусловленный личным опытом разработки web-приложений, для решения поставленной задачи, который впоследствии был утвержден разработчиками проекта O-GIS. В конечном итоге редактор композиций O-GIS был адаптирован под мобильные устройства. Модифицированный модуль удовлетворяет 8 требованиям, выдвинутым к конечному решению, проверка оставшихся 3 требований станет возможной после запуска системы в рабочем режиме.

Однако модифицированная система имеет недостаток: в случае, если десктопное устройство — сенсорное, некоторые функции, такие как измене-

ние порядка слоев геопространственных данных, будут недоступны пользователю, поскольку он будет получать не модифицированную ГИС. В дальнейшем такой недочет возможно будет исправить: существуют специализированные сервисы, которые могут различать свойства устройства, с которого зашел пользователь. Их подключение в систему O-GIS разрешит проблему: сенсорные устройства (не только мобильные, но и десктопные) будут получать модифицированную систему, хорошо функционирующую на сенсорных компьютерах.

Стоит отметить, что модуль O-GIS для работы с геопространственными данными не единственный в системе. Проект O-GIS разрабатывался как узкоспециализированная социальная сеть, поэтому содержит страницы регистрации, авторизации пользователей, профили участников проекта, элементы взаимодействия пользователей — почта, совместные проекты и т. п. Перечисленные модули O-GIS также не адаптированы под мобильные устройства, поэтому следующей задачей развития ГИС может стать модификация остальных частей системы.

Заключение

Автором данной выпускной квалификационной работы была решена поставленная задача: адаптирование модуля открытой многопользовательской географической информационной системы O-GIS под мобильные устройства. Автор приобрел незаменимый опыт разработки программных продуктов, в частности: планирование этапов разработки, проектирование архитектуры web-приложения, сравнение инструментов разработки и выбор наиболее подходящего для решаемой задачи, применение популярных инструментов разработки на практике, знакомство с web-технологиями в области геоинформационных систем. Дополнительно, автором были получены знания о методах работы с геопространственными данными, такими как геокодирование, перепроецирование, растровая алгебра, растровая реклассификация.

Список литературы

1. Официальное описание библиотеки отображения картографических данных OpenLayers 2. http://openlayers.org/two/
2. Официальный список выпусков библиотеки OpenLayers 2 на GitHub. https://github.com/openlayers/ol2/releases
3. Официальное описание проекта библиотеки отображения картографических данных OpenLayers 4. http://openlayers.org/
4. Официальное описание библиотеки отображения картографических данных LeafLet. http://leafletjs.com/
5. Информация о проектах разработчика библиотеки LeafLet Владимира Агафонкина. https://github.com/mourner
6. Описание библиотеки Google Maps в сетевой энциклопедии Википедия. https://ru.wikipedia.org/wiki/ Карты_Google
7. Официальное описание библиотеки отображения картографических данных Google Maps. https://developers.google.com/maps/documentation/javascript/tutorial
8. Официальный онлайн-реестр картографических проекций. http://www.epsg.org/
9. Описание проекции Меркатора в сетевой энциклопедии Википедия. https://ru.wikipedia.org/wiki/Проекция_Меркатора
10. Описание библиотеки LeafLet в сетевой энциклопедии Википедия. https://ru.wikipedia.org/wiki/Leaflet
11. Информация о пространственных данных сервиса «Космосиники». http://geomixer.ru/index.php/ru/docs/manual/datum
12. Информация о пространственных данных сервиса «Яндекс.Карты». https://tech.yandex.ru/maps/doc/theory/concepts/coordinates-docpage/

13. Официальное описание библиотеки отображения картографических данных Polymaps. http://polymaps.org/
14. Загрузочные файлы библиотеки OpenLayers 4. http://openlayers.org/download/
15. Загрузочные файлы библиотеки LeafLet. http://leafletjs.com/download.html
16. Сервис совместного создания и свободного распространения карт мира OpenStreetMap. http://openstreetmap.ru/
17. Список разработчиков библиотеки OpenLayers 4. https://github.com/openlayers/openlayers/graphs/contributors
18. Список запросов сообщества на модификацию компонентов библиотеки OpenLayers 4. https://github.com/openlayers/openlayers/pulls
19. Список обучающих материалов для изучения библиотеки OpenLayers 4. https://openlayers.org/en/latest/doc/
20. Список примеров отображения картографических данных библиотеки OpenLayers 4. https://openlayers.org/en/latest/examples/index.html
21. Список примеров отображения картографических данных библиотеки LeafLet. http://leafletjs.com/examples.html
22. Описание библиотеки OpenLayers 4 на GitBook. http://openlayers.org/workshop/en/index.html
23. Официальное описание сервера Apache. https://httpd.apache.org/
24. Официальное описание картографического сервера GeoServer. http://geoserver.org/
25. Официальное описание базы данных PostGIS. http://www.postgis.net/
26. Официальное описание PHP-фреймворка Symfony. http://symfony.com/
27. Официальное описание JavaScript библиотеки jQuery. https://jquery.com/

28. Описание концепции Model-View-Controller в сетевой энциклопедии Википедия. https://ru.wikipedia.org/wiki/Model-View-Controller
29. Описание технологии ORM в сетевой энциклопедии Википедия. https://ru.wikipedia.org/wiki/ORM
30. Официальное описание инструмента статического анализа ESlint. http://eslint.org/docs/about/
31. Официальное описание инструмента статического анализа JSlint. http://www.jslint.com/help.html
32. Информация о создателе JSlint инструмента и формата JSON — Крокфорде Дугласе в сетевой энциклопедии Википедия. https://ru.wikipedia.org/wiki/Крокфорд,_Дуглас
33. Официальное описание инструмента статического анализа JSHint. http://jshint.com/about/
34. Описание инструмента статического анализа JSHint в сетевой энциклопедии Википедия. https://en.wikipedia.org/wiki/JSHint
35. Описание языка программирования JavaScript в сетевой энциклопедии Википедия. https://ru.wikipedia.org/wiki/JavaScript
36. Официальное описание инструмента Babel. https://babeljs.io/docs/usage/api/
37. Официальное описание инструмента Webpack. https://webpack.js.org/concepts/
38. Официальное описание инструмента React. https://facebook.github.io/react/
39. Официальное описание инструмента React Native. https://facebook.github.io/react-native/
40. Официальное описание инструмента Create React App. https://github.com/facebookincubator/create-react-app

41. Официальное описание спецификации Material Design. https://material.io/guidelines/
42. Официальное описание библиотеки Material-UI. http://www.material-ui.com/
43. Описание подхода REST в сетевой энциклопедии Википедия. https://ru.wikipedia.org/wiki/REST
44. Описание заголовка HTTP запроса User-Agent в сетевой энциклопедии Википедия. https://ru.wikipedia.org/wiki/User_Agent

Приложение

Точка входа в клиентскую часть модифицированного модуля системы O-GIS:

```
import React from 'react';
import ReactDOM from 'react-dom';
import injectTapEventPlugin from 'react-tap-event-plugin';
import MuiThemeProvider from 'material-ui/styles/MuiThemeProvider';

import GisDrawer from './components/GisDrawer';
import createMap from './map';
import ready from './env-ready';
import './index.css';

const App = () => (
  <MuiThemeProvider>
    <GisDrawer />
  </MuiThemeProvider>
);

injectTapEventPlugin();

ready.then(() => {
  ReactDOM.render(
    <App />,
    document.getElementById('root'),
  );

  createMap();
});
```

React-компонент, отображающий редактор композиций O-GIS:

```
import React from 'react';
import Drawer from 'material-ui/Drawer';
import { List, ListItem } from 'material-ui/List';
import IconButton from 'material-ui/IconButton';
import SettingsSvg from 'material-ui/svg-icons/action/settings';
import RefreshSvg from 'material-ui/svg-icons/navigation/refresh';
import AddSvg from 'material-ui/svg-icons/content/add-box';
import SaveSvg from 'material-ui/svg-icons/device/sd-storage';
import SaveAsSvg from 'material-ui/svg-icons/content/save';
import FavoritesSvg from 'material-ui/svg-icons/toggle/star';
import MathSvg from 'material-ui/svg-icons/image/gradient';
import ExitSvg from 'material-ui/svg-icons/action/exit-to-app';
import EditSvg from 'material-ui/svg-icons/image/edit';
import Divider from 'material-ui/Divider';
import Checkbox from 'material-ui/Checkbox';
import Visibility from 'material-ui/svg-icons/action/visibility';
import VisibilityOff from 'material-ui/svg-icons/action/visibility-off';

import getLayersInfo from '../../../CompositionEditor';

const styles = {
  root: {
    position: 'absolute',
    marginTop: 60,
  },
  svgIcon: {
    color: 'lightseagreen',
    hoverColor: 'teal',
  },
  iconButton: {
    width: 35,
    zIndex: 2,
  },
  settingSvgIcon: {
    color: 'rgba(0,60,136,.5)',
    hoverColor: 'rgb(76,118,171)',
  },
  tooltipIconButton: {
    marginRight: 35,
    marginTop: -55,
  },
};

const iconButtonSetting = {
  touch: true,
  style: { ...styles.iconButton },
  disableTouchRipple: true,
};

export default class DrawerUndockedExample extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      open: false,
      layersList: getLayersInfo();
    };
  }

  handleToggle = () => this.setState({ open: !this.state.open });
```

```
handleClose = () => this.setState({ open: false });

render() {
  return (
    <div style={{ ...styles.root }}>
      <IconButton
        onTouchTap={this.handleToggle}
        tooltip="Настройки"
        tooltipPosition="bottom-right"
        touch
      >
        <SettingsSvg {...styles.settingSvgIcon} />
      </IconButton>
      <Drawer
        docked={false}
        width={257}
        open={this.state.open}
        onRequestChange={open => this.setState({ open })}
      >
        <IconButton
          tooltip="Добавить слой"
          tooltipPosition="bottom-right"
          {...iconButtonSetting}
        >
          <AddSvg {...styles.svgIcon} />
        </IconButton>
        <IconButton
          tooltip="Операции со слоями"
          tooltipPosition="bottom-right"
          {...iconButtonSetting}
        >
          <MathSvg {...styles.svgIcon} />
        </IconButton>
        <IconButton
          tooltip="Сбросить изменения"
          tooltipPosition="bottom-right"
          {...iconButtonSetting}
        >
          <RefreshSvg {...styles.svgIcon} />
        </IconButton>
        <IconButton
          tooltip="Сохранить"
          tooltipPosition="bottom-right"
          {...iconButtonSetting}
        >
          <SaveSvg {...styles.svgIcon} />
        </IconButton>
        <IconButton
          tooltip="Сохранить как"
          tooltipPosition="bottom-center"
          {...iconButtonSetting}
        >
          <SaveAsSvg {...styles.svgIcon} />
        </IconButton>
        <IconButton
          tooltip="Добавить в избранное"
          tooltipPosition="bottom-left"
          {...iconButtonSetting}
        >
          <FavoritesSvg {...styles.svgIcon} />
        </IconButton>
        <IconButton
          tooltip="Покинуть редактор"
```

```
            tooltipPosition="bottom-left"
            {...iconButtonSetting}
          >
            <ExitSvg {...styles.svgIcon} />
          </IconButton>
          <List>
            <Divider />
            {Object.entries(this.state.layersList)
              .map(([id, { name, checked }]) =>
                (<ListItem
                  key={id}
                  primaryText={name}
                  leftCheckbox={
                    <Checkbox
                      checked={checked}
                      checkedIcon={<Visibility />}
                      uncheckedIcon={<VisibilityOff />}
                    />
                  }
                  rightIconButton={
                    <IconButton
                      touch
                      tooltip="Правка"
                      tooltipPosition="bottom-left"
                      tooltipStyles={{ ...styles.tooltipIconButton }}
                    >
                      <EditSvg {...styles.svgIcon} />
                    </IconButton>
                  }
                />),
            )}
          </List>
        </Drawer>
      </div>
    );
  }
}
```

React-компонент, визуализирующий географическую карту:

```
import { Map, View, layer, source, control } from 'openlayers';
import 'openlayers/dist/ol.css';

import getLayers from '../../../CompositionEditor';
import getParams from '../../../CompositionEditor';

export default () => new Map({
  layers: [
    getLayers(),
  ],
  target: 'map',
  controls: control.defaults({
    attributionOptions: ({
      collapsible: false,
    }),
  }),
  view: new View(getParams()),
});
```

С остальным кодом модифицированного проекта O-GIS можно ознакомиться на странице репозитория GitHub: https://github.com/oonsamyi/o-gis.